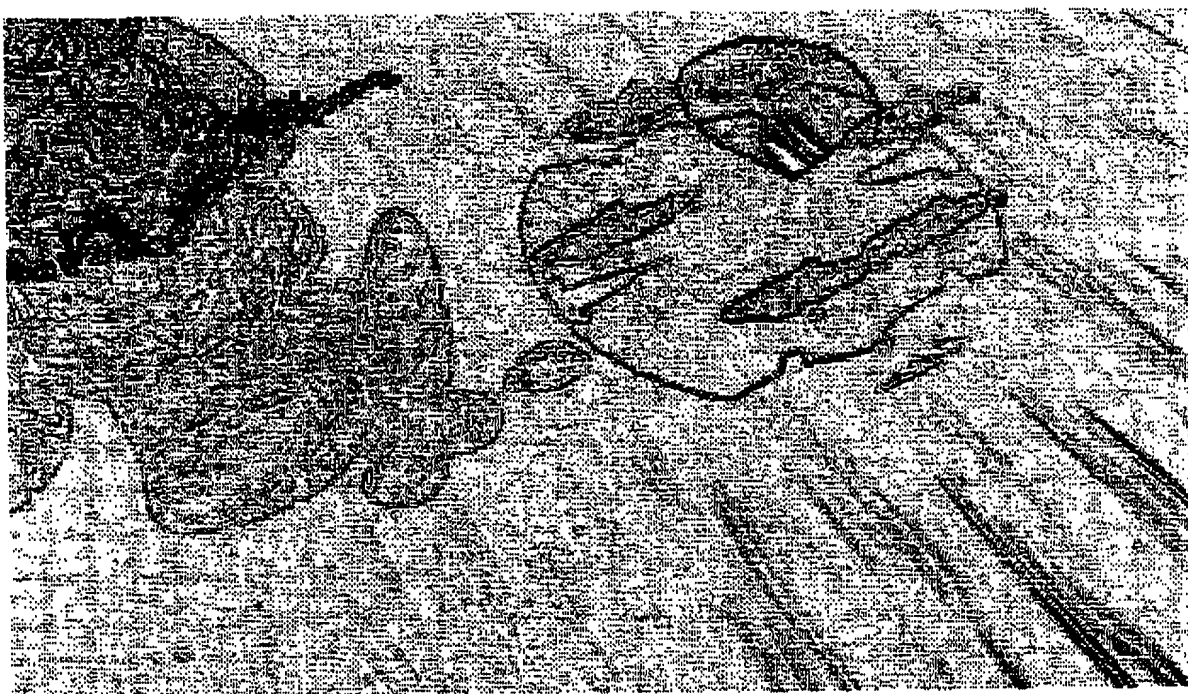


FIG.1



FOUO-111250

FIG.2

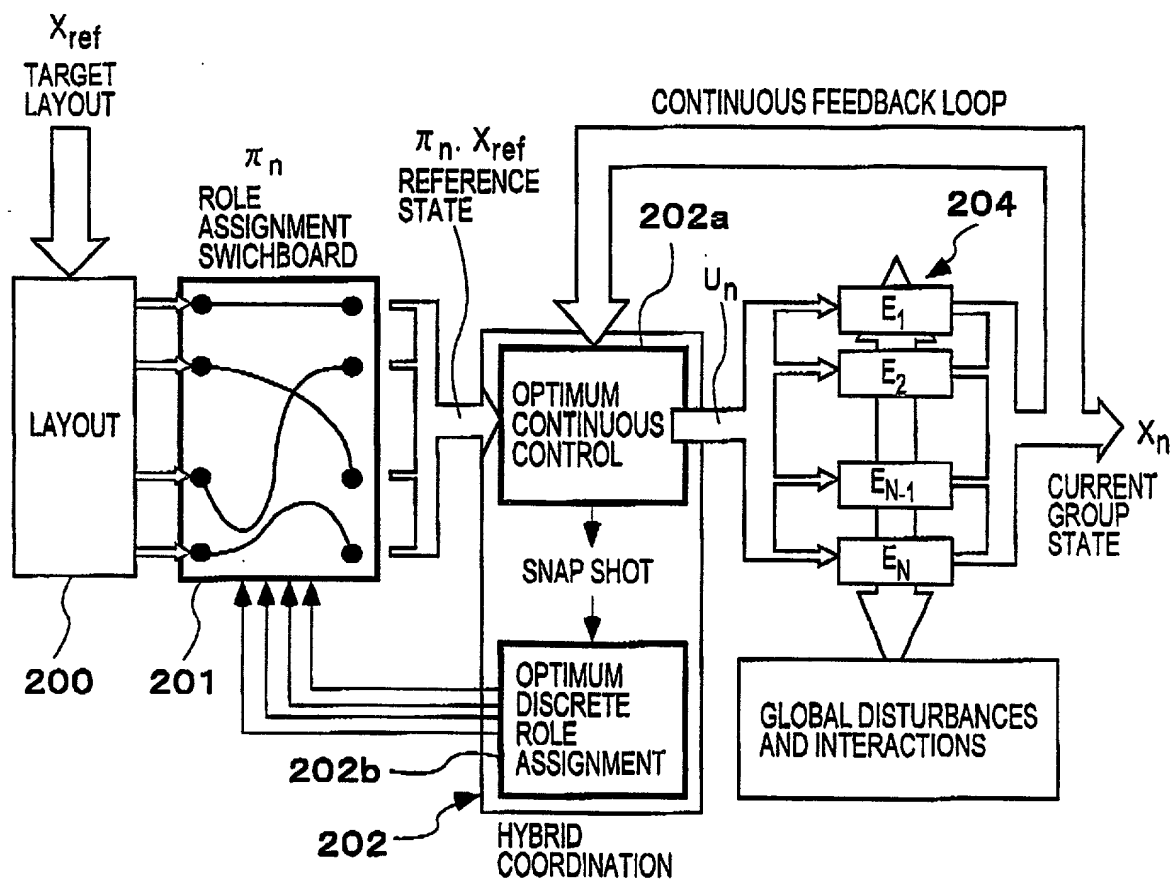


FIG.3

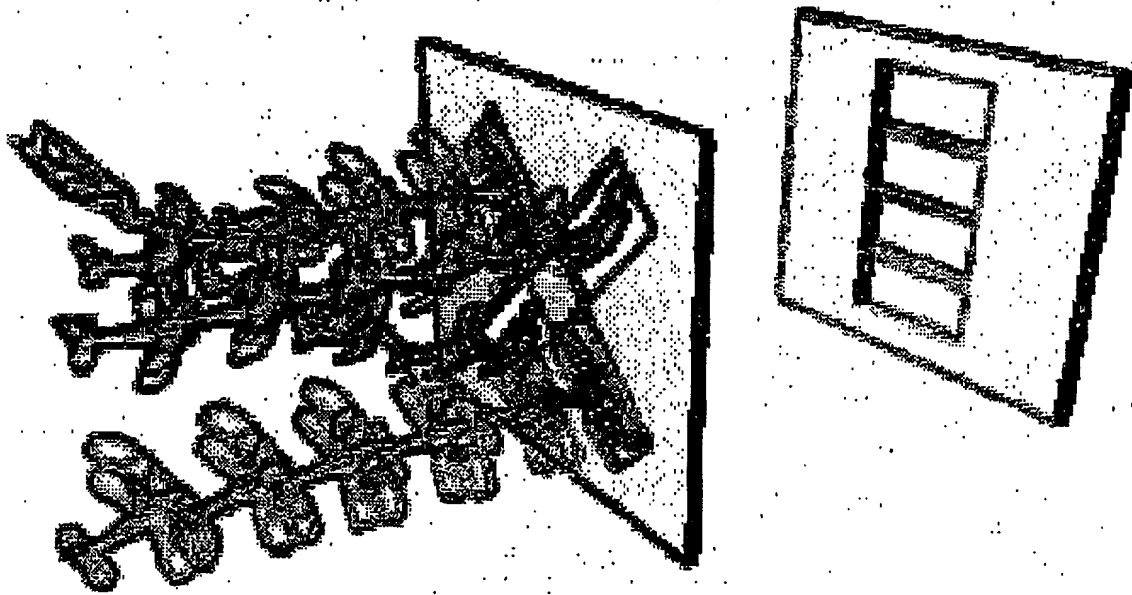


FIG. 4

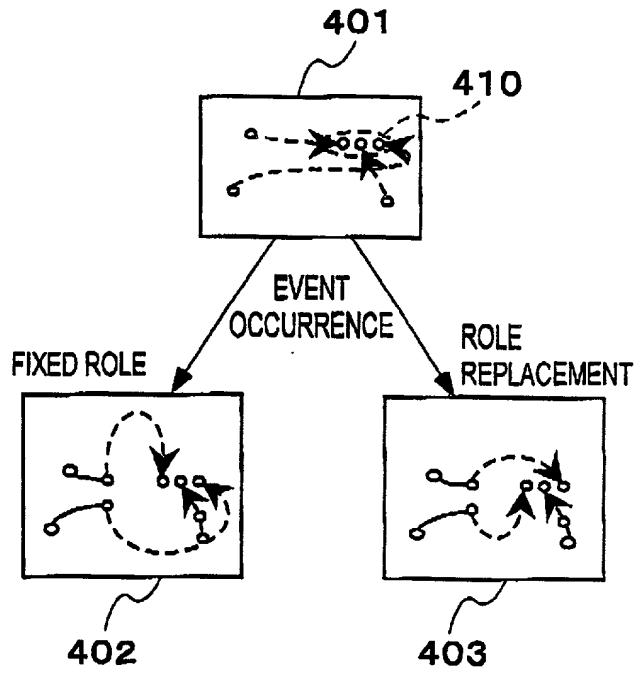


FIG.5

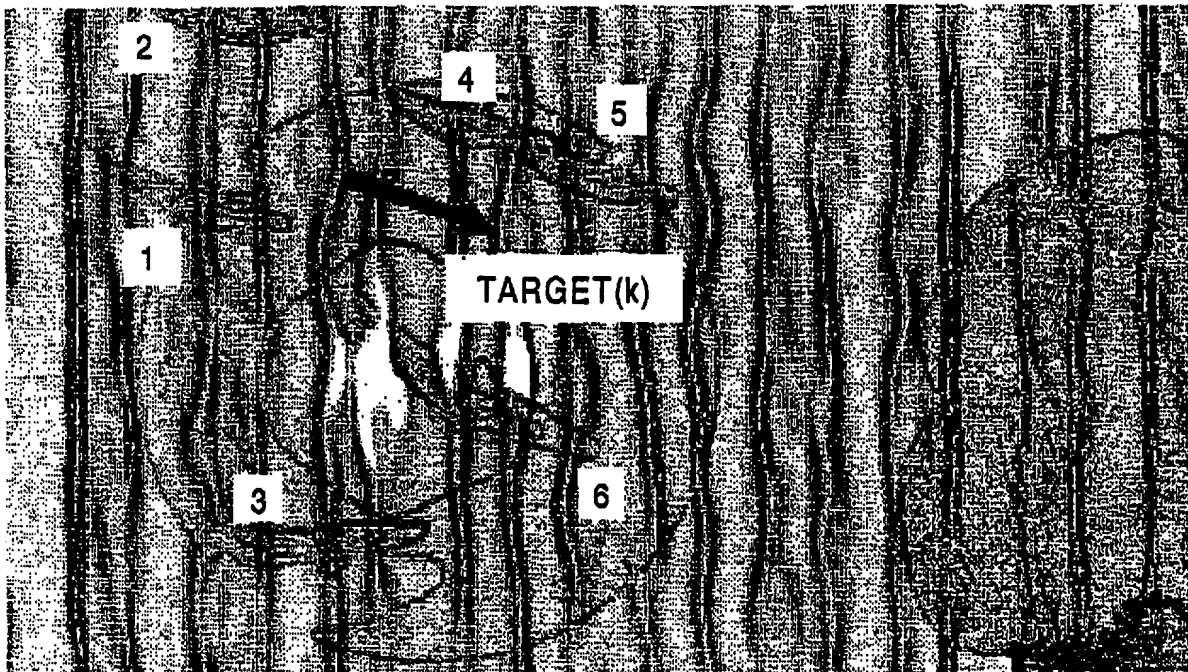


FIG.6

RUN-TIME COORDINATION ALGORITHM (A1)EXERT CONTINUOUS MULTIVARIABLE CONTROLS (ROLE ASSIGNMENTS DEFINED BY π_n)

$$1. \quad U_n \leftarrow -G[X_n - \pi_n \cdot X_{ref}]$$

DYNAMIC SIMULATION STEP

$$2. \quad X_{n+1} \leftarrow f(X_n, U_n)$$

$$3. \quad J_n \leftarrow (X_n - \pi_n \cdot X_{ref})^T S_n (X_n - \pi_n \cdot X_{ref})$$

SORT ENTITIES ACCORDING TO THEIR CONTRIBUTION TO OVERALL COST

4 For each entity j do

$$5 \quad X_n^{-j} \leftarrow [X_n^1, X_n^2, \dots, j^{\text{th}} \text{ coordinate of } (\pi_n X_{ref}), \dots, X_n^N]$$

$$6 \quad J_j(n) \leftarrow (X_n^{-j} - \pi_n \cdot X_{ref})^T S_n (X_n^{-j} - \pi_n \cdot X_{ref})$$

7 Endfor

8 Sort entities by decreasing $J_j(n)$

LOOK FOR OPPORTUNISTIC ROLE SWAPS IN K WORST PERFORMERS

9 For $K!$ permutations $\pi(k)$ of K candidates do

$$10 \quad J_n(\pi(k)) \leftarrow (X_n - \pi(k) \cdot X_{ref})^T S_n (X_n - \pi(k) \cdot X_{ref})$$

11 if $(J_n^* > |J_n(\pi(k)) + \Delta_{min}|)$ then

$$12 \quad J_n^* \leftarrow J_n(\pi(k))$$

$$13 \quad k^* \leftarrow k$$

14 endif

15 endfor

UPDATE ROLE ASSIGNMENT SWITCHBOARD

$$16 \quad \pi_{n+1} \leftarrow \pi(k)^*$$

1. FEED-FORWARD GAINS (OR INTEGRAL TERMS) ENSURE DISTURBANCE REJECTION.

2. THRESHOLD PREVENTS FLICKERING DUE TO NON-LINEAR DYNAMICS.

3. PERMUTATIONS ARE COMPUTED BEFOREHAND (SEE OFFLINE ALGORITHM IN FIG.15).

4. ROLE ASSIGNMENT IS REPRESENTED BY MATRIX f FOR CONVENIENCE, YET IMPLEMENTATION RELIES ON A SIMPLE INDIRECTION RATHER THAN MATRIX MULTIPLICATION.

FIG.7A

PERFORMANCE INDEX

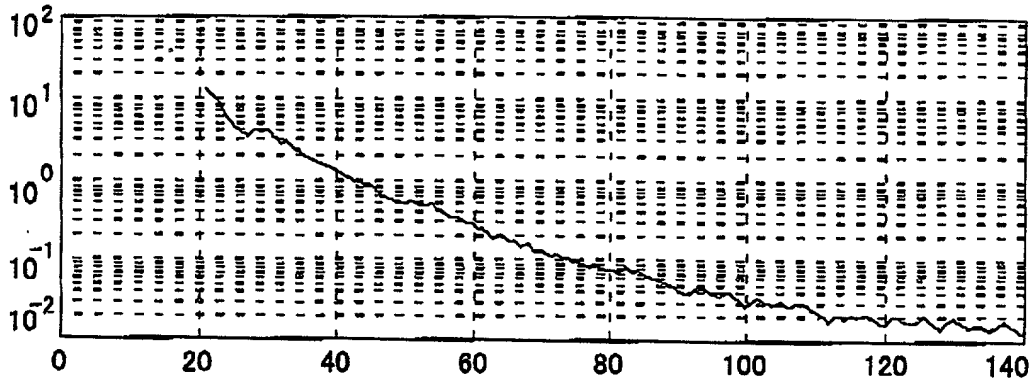


FIG.7B

NUMBER OF LOCAL INTERACTIONS

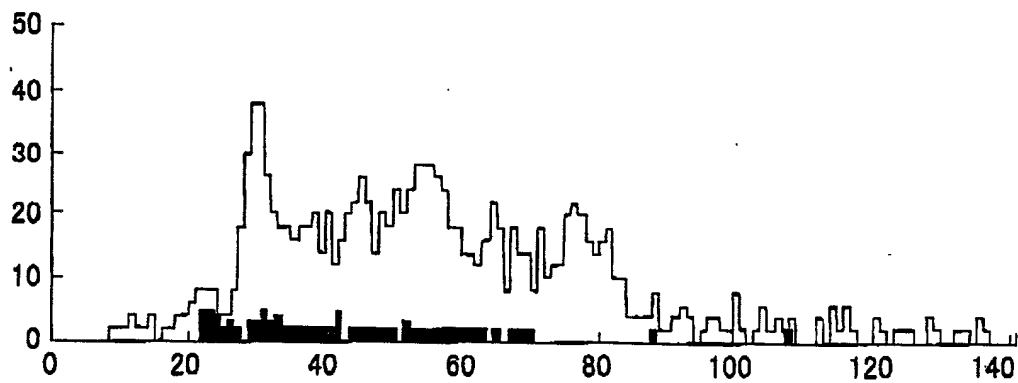


FIG.8

OPTIMIZED VERSION FOR BLOCK-DIAGONAL SYSTEMS (A2)

```

1       $U_n \leftarrow -G[X_n - \pi_n \cdot X_{ref}]$ 
2       $X_{n+1} \leftarrow f(X_n, U_n)$ 
3      For  $j=1$  to  $N$  do
4           $\delta J(i, j) \leftarrow [X_n^j - (\pi_n \cdot X_{ref})^j]^T S^{entity} [X_n^j - (\pi_n \cdot X_{ref})^j]$ 
5      endfor
6      Sort entities by decreasing  $\delta J(i, I)$  and select first  $K$  ones
7      for  $k=1$  to  $K$  do
8          for  $p=k$  to  $K$  do
9               $\delta J(k, p) \leftarrow [X_n^k - X_{ref}^p]^T S^{entity} [X_n^k - X_{ref}^p]$ 
10         endfor
11     endfor

```

LOOK FOR OPPORTUNISTIC ROLE SWAPS IN K WORST PERFORMERS

```

12     For  $K!$  permutations  $\pi(k)$  of  $K$  candidates do
13          $J_{\omega}(\pi(k)) \leftarrow \sum_{p=0}^N \delta J(p, -X_{ref}(p, \pi(k)) \text{ or } target(p, \pi(k)))$ 
14         if  $(J_{\omega} > |J_{\omega}(\pi(k)) + \Delta_{min}|)$  then
15              $J_{\omega}^* \leftarrow J_{\omega}(\pi(k))$ 
16              $k^* \leftarrow k$ 
17         endif
18     endfor
19      $\pi_{n+1} \leftarrow \pi(k^*)$ 

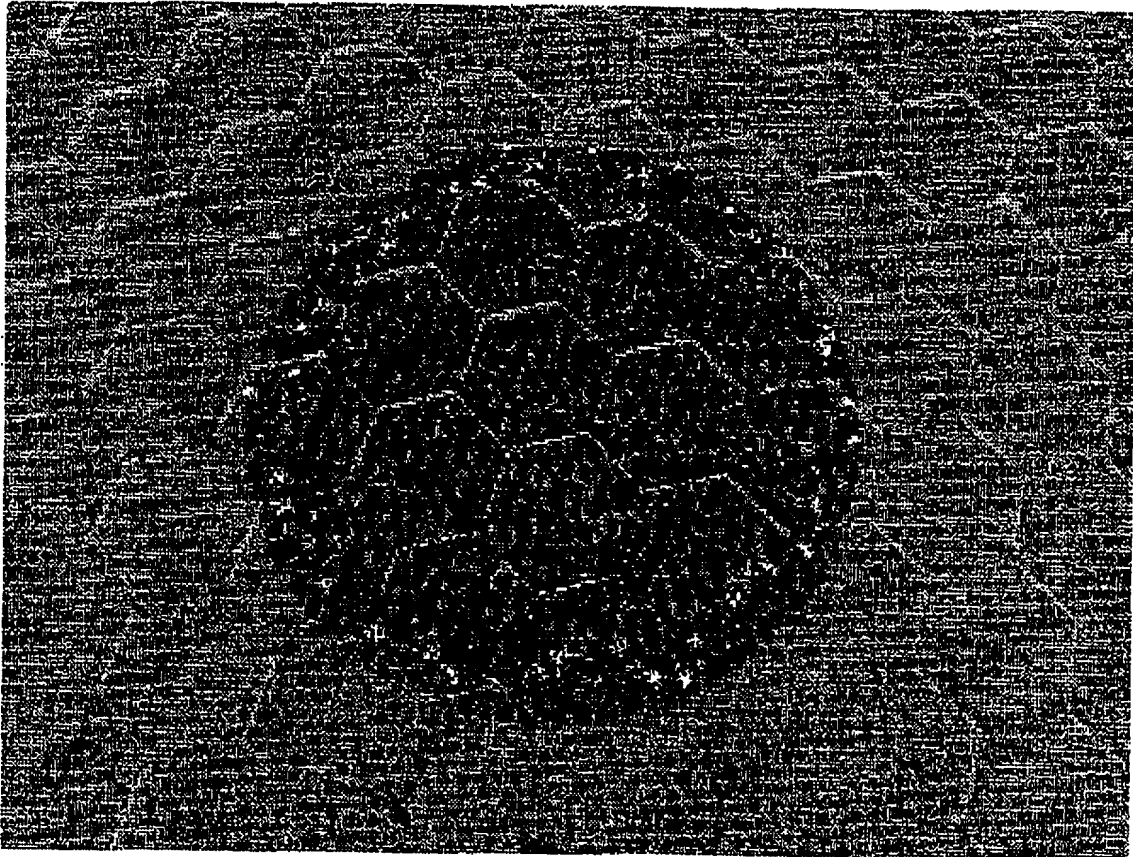
```

(1) SORTING CANDIDATES FOR ROLE SWAPPING ON A SIMPLE DISTANCE-TO-TARGET CRITERION
CORRESPONDS TO REPLACING S^{entity} BY THE IDENTITY MATRIX.

FIG.9

MEMBER	ALGORITHM	N	n_e	Nn_e	COORDI- NATION	TOTAL/ STEP	REAL-TIME
FISH	A2	6	6	36	8KFLOPS	16KFLOPS	YES
AIRCRAFT	A2	6	11	66	14.6KFLOPS	23KFLOPS	YES
BEE	A1	50	4	20	17MFLOPS	16.8MFLOPS	RESTRICTIVE
BEE	A1	100	4	400	67MFLOPS	68MFLOPS	NO

FIG.10



2025.07.10 14:00:00

FIG.11

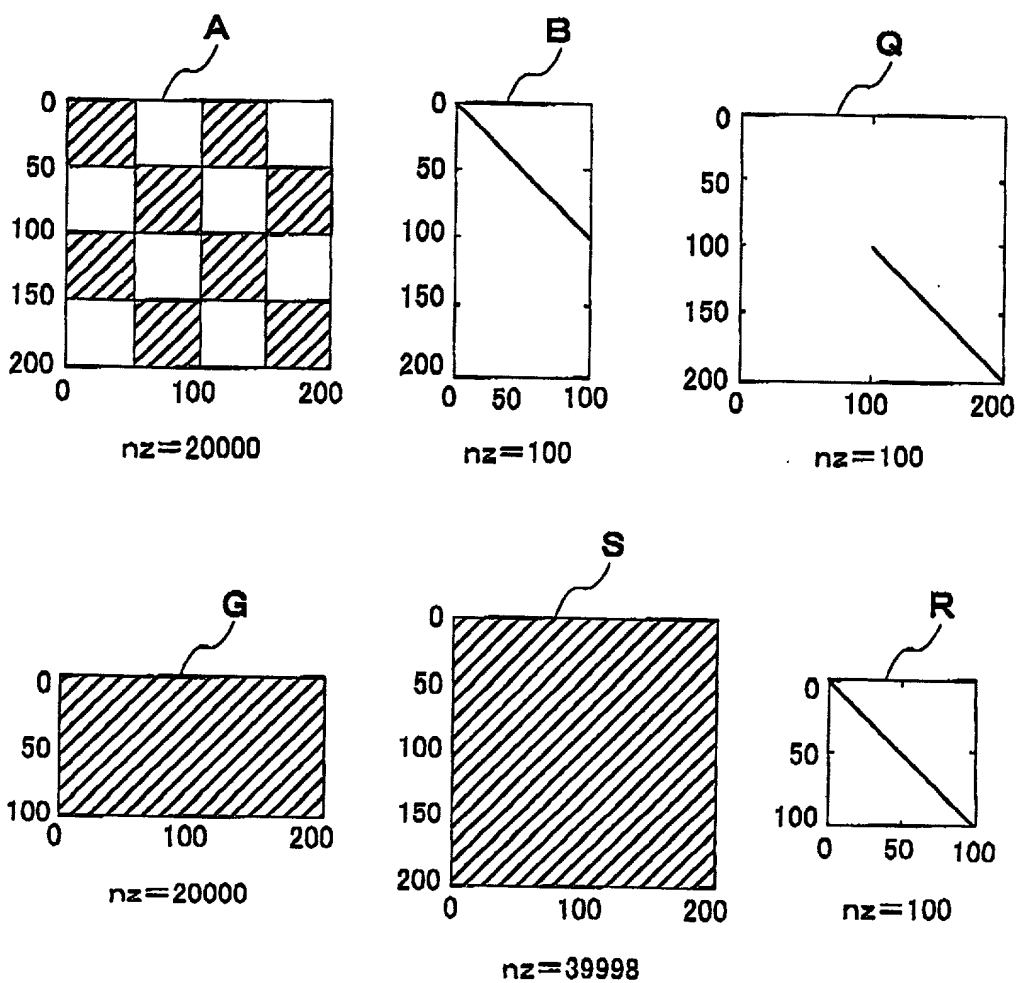


FIG. 12B

NUMBER OF FRAMES

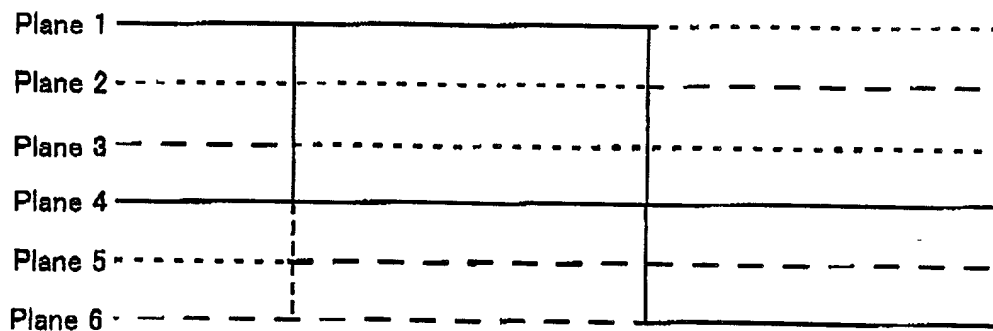


FIG.13A

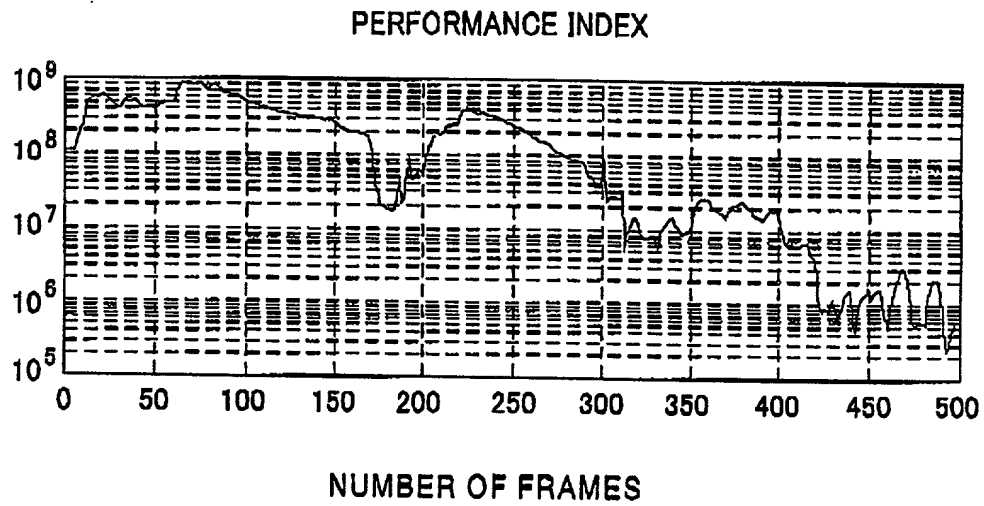


FIG.13B

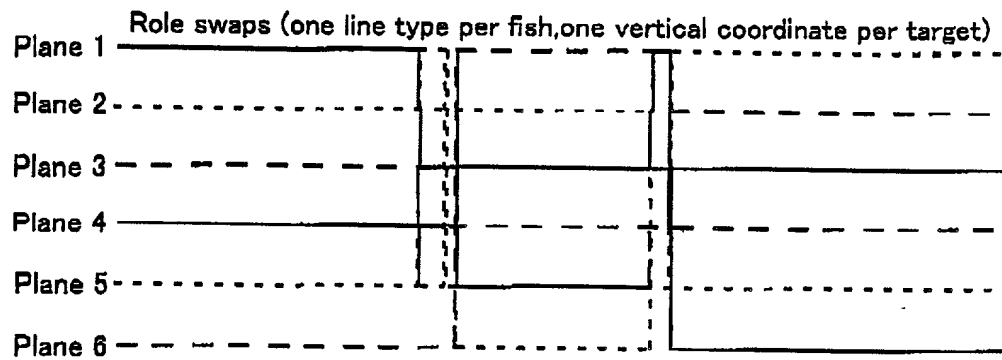


FIG.14

MULTIVARIABLE LQ CONTROL SYSTHESIS

- 1 GIVEN THE DYNAMIC SYSTEM A, B AND PERFORMANCE INDEX MATRICES Q, R, FORM THE HAMILTONIAN MATRIX DEFINED AS:

$$H = \begin{bmatrix} B + AR^{-1}B^TA^TQ & -BR^{-1}B^TA^T \\ -A^TQ & A^T \end{bmatrix}$$

- 2 COMPUTE EIGENVALUES λ AND EIGENVECTORS W OF H, COLLECTING ALL COUPLES SUCH THAT THE EIGENVALUE'S MODULUS IS SMALLER THAN ONE (STABLE POLE OF THE CLOSED-LOOP SYSTEM)

- 3 COMPUTE $S_{\infty} = \lambda W^{-1}$

- 4 COMPUTE THE OPTIMAL MATRIX K GAIN AS

$$G = [R + B^TS_{\infty}B]^{-1}B^TS_{\infty}A$$

FIG.15

Function PermutationArray=Permute(vector)

```
1      n=length(vector)
2      if(n==1) then
3          PermutationArray=vector
4      else
5          PermutationArray=NULL array
6          Height=0
7          for k=1 to n do
8              sub-vector(k)=ShrinkVector(vector, k)
9              SubArray=Permute(sub-vector)
10             for i=1 to (n-1)! do
11                 for j=1 to n-1 do
12                     PermutationArray [i+Height][j]=SubArray[i][j]
13                 endfor
14                 PermutationArray [i+Height][n]=vector [k]
15             endfor
16             Height=Height+(n-1)!
17         endfor
18     endif
19     return PermutationArray
```

FIG.16

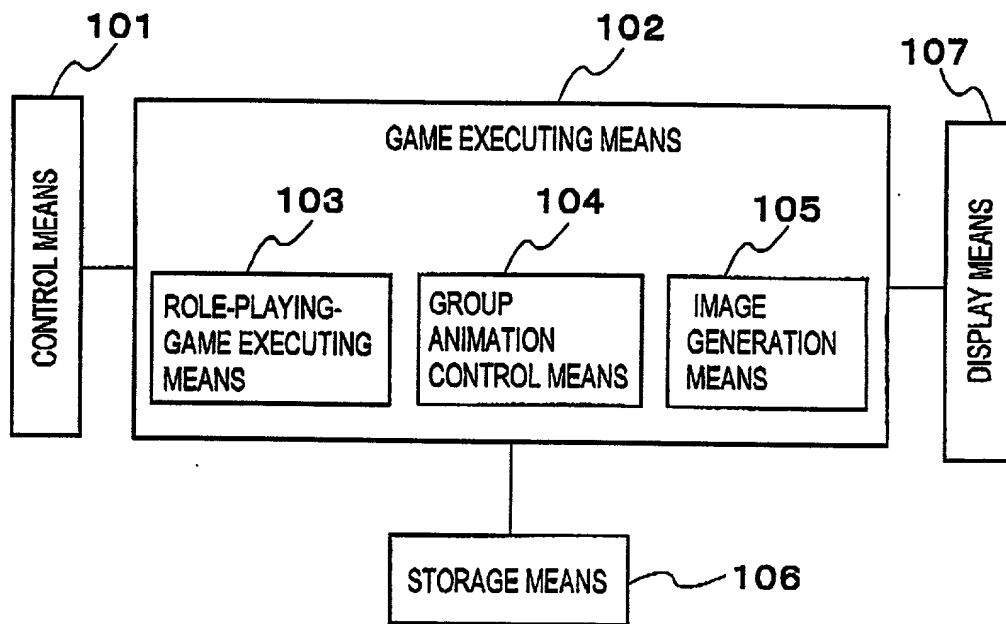


FIG. 17

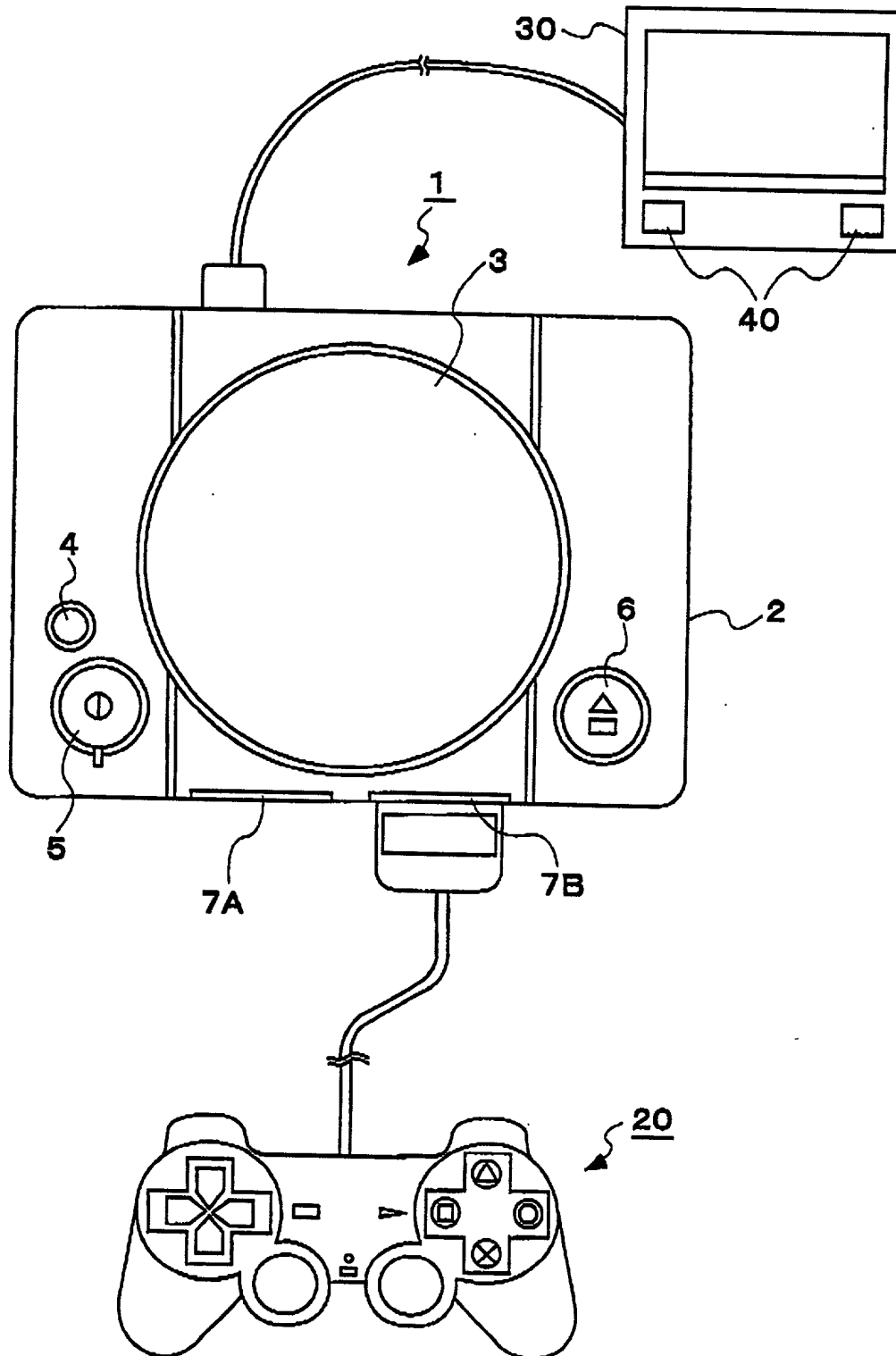


FIG.18

